UNITED STATES DISTRICT COURT

NORTHERN DISTRICT OF CALIFORNIA

SAN JOSE DIVISION

NAZOMI COMMUNICATIONS, INC., Plaintiff, v. NOKIA CORPORATION et al., Defendants.	Case No. C-10-04686 RMW (related) **ORDER CONSTRUING CLAIMS OF U.S. PATENT NOS. 7,080,362 AND 7,225,436**
NAZOMI COMMUNICATIONS, INC., Plaintiff, v. SAMSUNG TELECOMMUNICATIONS AMERICA, L.L.C. et al., Defendants.	

On November 28, 2012, the court held a claim construction hearing for the purpose of construing disputed terms as described below. At issue in this claim construction is the scope of two of Nazomi Communications Inc.'s patents. Nazomi generally argues for broad dictionary definitions of the terms used in the patents, while the defendants insist that the patent specifications disclose and define a more limited invention. For the reasons explained below, the court generally agrees with the defendants' narrower proposed constructions of the disputed terms.

I. BACKGROUND

Nazomi brings this patent infringement action against various technology companies, alleging infringement of claims 1, 15, 17, 22, 26, 48, 66, 67, 68, 69, and 70 of U.S. Patent No. 7,080,362 (the '362 patent), and claims 1, 5, 12, and 14 of U.S. Patent No. 7,225,436 (the '436 patent) both continuations of U.S. Patent No. 6,332,215 (the '215 parent patent). The '362 patent and '215 parent patent, both titled "Java Virtual Machine Hardware for RISC and CISC Processors," and the '436 patent, titled "Java Hardware Acceleration Using Microcode Engine," all relate to technology which uses hardware for accelerating the execution of stack-based instructions like Java bytecodes. The specifications for the '215 patent and '362 patent are essentially identical. The '436 patent also has the same disclosure but adds additional features. In this case, Nazomi claims that many of defendants' consumer products infringe its patents by using microprocessors with specialized hardware to accelerate the execution of Java programs.

A. The Technology

This case concerns technology for processing different types of machine code, or "instruction sets," within the hardware of a computer. In particular, the patents relate to hardware to accelerate the processing of "stack-based" as opposed to "register-based" instructions. A "stack" and a "register" are different ways of storing data in memory and necessitate different types of associated instructions to manipulate that memory in order to execute a program. Java bytecodes and other stack-based instructions will not operate on a register-based system unless the system first translates them to refer to registers instead of a stack or otherwise processes them.

On a typical computer with a register-based processor, a program is just a series of instructions stored as 1s and 0s (most systems are 32-bits, which means a single instruction is made up of 32 1s and 0s). When a computer executes a program, the instructions that make up the program go through the well-defined process of the Central Processing Unit (CPU) pipeline known as the "fetch-decode-execute" cycle. First, the processor fetches the instruction in memory. Second, a decode unit converts the 1s and 0s of the instruction into hundreds if not thousands of control signals. Third, these control signals manipulate the registers and other logic gates in the processor to perform the data manipulation or mathematical calculation indicated by the instruction. Finally, the results of the manipulation are written back into a register as needed. *See* Dr. Babb Decl. at ¶ 11 (filed under seal); Decl. of Stephen Steele at ¶¶ 22-28, dkt. no. 405 ex. 3.[1]

B. The Patented Invention

The patents relate to a method of accelerating the execution of stack-based instructions on register-based processors. *See* '362 and '436 patents, Background and Summary of the Invention. The prior art teaches that the way to execute stack-based instructions, like a Java bytecode, on a register-based processor is by converting the stack-based instructions into register-based instructions in software. *Id.* The patents accelerate this process by adding new hardware elements that speed-up the conversion process and the execution of the instructions. *Id.*

Many register-based systems use a Java Virtual Machine—a software program—to translate Java bytecodes into register-based instructions. This approach provides flexibility, but slows overall execution speed. The patented invention "removes the bottleneck which previously occurred when the Java Virtual Machine is run in software . . . [by implementing] at least part of the Java Virtual Machine in hardware as the hardware Java accelerator." '362 Patent col.2 ll.10-17. The '436 patent builds on this general invention by disclosing a way of more efficiently handling interrupts—computer commands that require a shift from stack-based instructions to register-based instructions and then back again. '436 patent col.2 ll.5-59. The '436 patent also discloses a parallel decode unit that translates multiple bytecodes concurrently into a lesser number of register-based instructions. *Id.* col.6 ll.20-25.

[1] The remainder of this Order cites to the docket for case number C-10-04686, the first filed case, but identical copies of these documents were also filed in C-10-05545 as well.

C. Procedural History

Courts have previously considered and construed terms in patents related to Nazomi's alleged inventions. In *Nazomi Communications, Inc. v. Arm Holdings, PLC* ("*Nazomi 2002*" to distinguish it from the present cases), Nazomi and ARM disputed whether ARM's Jazelle Revision 3 design infringed the '215 parent patent. *See* No. C-02-02521-JF (N.D. Cal. 2002). Initially, the district court construed "instruction" and granted summary judgment of noninfringement based upon its construction. *Nazomi 2002 I*, 2003 WL 24054503 (N.D. Cal. Sept. 30, 2003). The Federal Circuit vacated and remanded, finding that the court's analysis was inadequate. *Nazomi 2002 II*, 403 F.3d 1364 (Fed. Cir. 2005). Thereafter, the district court conducted a detailed analysis of the term "instruction." *Nazomi 2002 III*, 2006 WL 2578374 (N.D. Cal. Sept. 6, 2006). Nazomi objected to the court's construction and appealed to the Federal Circuit, but conceded that under the district court's construction of "instruction," Jazelle Revision 3 based processors "would not infringe the '215 parent patent, literally or under the doctrine of equivalents." *Nazomi 2002*, Resp. to Def.'s Summ. J. Mot., dkt. no. 259 at 3 (Jan. 19, 2007). The Federal Circuit affirmed the district court's construction, which ended the case. *Nazomi 2002 IV*, 266 F. App'x 942, 942 (Fed. Cir. 2008).

In this case, Nazomi brings claims based upon the children of the '215 parent patent against consumer product manufacturers that make products using processors incorporating ARM's Jazelle Revision 3 design. ARM has intervened on behalf of defendants, and in addition to arguing for particular claim constructions for the disputed terms addressed in this Order, it has filed for summary judgment on behalf of all of the defendants.

II. LEGAL STANDARD

Claim construction is exclusively within the province of the court. *Markman v. Westview Instruments, Inc.*, 517 U.S. 370, 387 (1996). "It is a 'bedrock principle' of patent law that 'the claims of a patent define the invention to which the patentee is entitled the right to exclude.'" *Phillips v. AWH Corp.*, 415 F.3d 1303, 1312 (Fed. Cir. 2005) (en banc) (internal citation omitted). Claim terms "are generally given their ordinary and customary meaning," defined as "the meaning . . . the term would have to a person of ordinary skill in the art in question . . . as of the effective filing date of the patent application." *Id.* at 1313 (internal citation omitted). The skilled artisan reads the claim term "in the context of the entire patent . . . including the specification." *Id. See also Multiform Desiccants, Inc. v. Medzam, Ltd.*, 133 F.3d 1473, 1477 (Fed. Cir. 1998). In determining the meaning of a disputed claim limitation, the intrinsic evidence, including the claim language, written description, and prosecution history, is the most significant. *Phillips*, 415 F.3d at 1315-17. The court reads claims in light of the specification, which is "the single best guide to the meaning of a disputed term." *Id.* at 1315. Furthermore, "the interpretation to be given a term can only be determined and confirmed with a full understanding of what the inventors actually invented and intended to envelop with the claim." *Id.* at 1316 (quoting *Renishaw PLC v. Marposs Societa' per Azioni*, 158 F.3d 1243, 1250 (Fed. Cir. 1998)). The words of the claims must be understood as the inventor used them as revealed by the patent and prosecution history. *Id.*

Although extrinsic evidence is less significant than the intrinsic record, the court may also reference extrinsic evidence to "shed useful light on the relevant art." *Id.* at 1317 (quoting *C.R. Bard, Inc. v. U.S. Surgical Corp.*, 388 F.3d 858, 862 (Fed. Cir. 2004)). "[T]echnical dictionaries may provide [help] to a court 'to better understand the underlying technology' and the way in which one of skill in the art might use the claim terms. . . . Such evidence . . . may be considered if the court deems it helpful in determining 'the true meaning of language used in the patent claims.'" *Id.* at 1318 (internal citations omitted).

Nazomi asserts, and defendants do not dispute, that for the purposes of these patents, a person of ordinary skill in the art at the time of the invention would have either: (1) a bachelor of science in computer engineering, electrical engineering, or computer science with about three years of work experience in the field of computer architecture; or (2) a master's of science in computer engineering, electrical engineering, or computer science with about a year of work experience in the field of computer architecture. Pl.'s Opening Claim Construction Br. at 1, Dkt. No. 402 ("Pl.'s Br."). There is no term whose construction turns on the level of skill in the art.

The court construes the five disputed terms that the parties identified as the "most important" to the resolution of the case."[2] The first term, "instructions," was at issue in the prior Nazomi lawsuit, *Nazomi 2002*, involving the '215 parent patent. The term "instruction" (in the singular) is also disputed with respect to U.S. Patent No. 6,338,160 (the '160 patent), in Case No. C-10-05545 concurrently before this court. The court's construction of "instructions" significantly affects the construction of the other terms. In particular, the parties dispute whether the patent requires translation of stack-based instructions into register-based instructions. Jazelle 3 processors are capable of executing both stack- and register-based instructions, but do not translate stack-based instructions into register-based instructions. Instead, the Jazelle 3 processor takes stack-based instructions and converts them directly into control signals. These control signals then control the registers and execute logic to execute the instructions. Under Nazomi's proposed claim constructions, the '362 and '436 patents cover this conversion from stack-based instructions to register-based control signals. Defendants propose a narrower construction that does not cover their products.

III. CLAIM CONSTRUCTION

A. "Instructions"

Nazomi's Proposed Construction	Defendants' Proposed Construction
Program commands interpreted or compiled into machine language.	Either a stack-based instruction that is to be translated into a register-based instruction, or a register-based instruction that is input to the CPU pipeline.

The term "instructions" appears in every claim at issue. Nazomi proposes that the term means "program commands interpreted or compiled into machine language" while the defendants

[2] The parties actually dispute additional terms, but under the Patent Local Rules the parties must identify 10 terms that are the "most significant to the resolution of the case," Patent L.R. 4-3(c), and five of the terms identified are from U.S. Patent No. 6,338,160 (the '160 patent) and thus not discussed in this order.

argue it should be "either a stack-based instruction that is to be translated into a register-based instruction, or a register-based instruction that is input to the CPU pipeline." The defendants' proposal is the construction adopted by the district court in *Nazomi 2002* and affirmed by the Federal Circuit for the '215 parent patent, which is the parent of both the '362 and '436 patents, the two patents-at-issue here. As explained below, the court generally should interpret the terms in a family of patents consistently. Furthermore, collateral estoppel requires that this court adopt the same construction for "instructions" in the '362 and '436 patents that Judge Fogel and the Federal Circuit did for the '215 parent patent. Finally, the claims and specifications of the patents support defendants' construction.

1. The Claims Must Be Construed the Same in Related Patents

Whenever possible, courts must construe claim terms consistently within a patent. *Southwall Techs., Inc. v. Cardinal IG Co.*, 54 F.3d 1570, 1579 (Fed. Cir. 1995). Courts should also construe the same term consistently across related patents. *Mycogen Plant Sci., Inc. v. Monsanto Co.,* 252 F.3d 1306, 1311 (Fed. Cir. 2001), *vacated on other grounds*, 535 U.S. 1109 (2002); s*ee also Toshiba Corp. v. Lexar Media, Inc.,* No. C-02-5273 MJJ, 2005 WL 6217120 (N.D. Cal. Jan. 24, 2005) ("Where patents-in-suit share the same disclosures, common terms are construed consistently across all claims in both patents."); *PCTEL, Inc. v. Agere Sys., Inc.*, No. C-03-2474 MJJ, 2005 WL 2206683 (N.D. Cal. Sept. 8, 2005) (same).

In *Nazomi 2002*, the court determined the meaning of "instruction" with respect to the '215 parent patent. 2006 WL 2578374, *aff'd*, 266 Fed. Appx. 935. The district court construed "instruction" to mean "either a stack-based instruction that is to be translated into a register-based instruction, or a register-based instruction that is input to the CPU pipeline." *Id.* at *8. The court also held that an instruction must be upstream of the decode stage of the CPU pipeline and cannot be the control signal that is the output of the decode stage. *Id.* The disclosures in '215 parent patent are essentially identical to those in the '362 patent. *Compare* the '215 parent patent figs.1-7D and cols.1-7 *with* the '362 patent figs.1-7D and cols.1-7. Similarly, the '436 patent contains the '215 parent patent's disclosures as well as additional disclosures. *Compare* the '215 parent patent figs. 1-7D and cols.1-7 *with* the '436 patent figs.1-7D and cols.1-7. Because a court should construe the same term the same across related patents, the court finds that the same construction should apply to the '362 and '436 patents.

2. Collateral Estoppel

Similarly, collateral estoppel requires that "instructions" be construed the same here as it was in *Nazomi 2002*. In order to establish collateral estoppel:

> (1) the issue at stake must be identical to the one alleged in the prior litigation; (2) the issue must have been actually litigated [by the party against whom preclusion is asserted] in the prior litigation; and (3) the determination of the issue in the prior litigation must have been a critical and necessary part of the judgment in the earlier action.

Trevino v. Gates, 99 F.3d 911, 923 (9th Cir. 1996); s*ee also Bayer AG. v. Biovail Corp.*, 279 F.3d 1340, 1345 (Fed. Cir. 2002) (holding that the law of the regional circuit determines the standard for collateral estoppel). Collateral estoppel can apply to common issues in actions involving different but related patents. *See, e.g., Amgen, Inc. v. Genetics Inst., Inc.*, 98 F.3d 1328, 1329-32, (Fed. Cir. 1996) (affirming summary judgment where a prior case had decided issues about a different patent with the same specification). Nazomi argues that the issues at stake are not identical, but does not contest that the issues were actually litigated or that the determination was a necessary part of the judgment. *See* Pl.'s Responsive Claim Construction Br. 3-10, Dkt. No. 412 ("Pl.'s Reply"). Therefore, the only question is whether the issues at stake are identical.

Here the parties are asking the court to construe the same term already twice construed by other judges with respect to related patents. In *Nazomi 2002*, the Federal Circuit directed the district court to look to the specification for the inventor's intended meaning for "instructions," finding that the inventor defined "instructions" in an indirect manner. *Nazomi 2002 II*, 403 F.3d at 1369. On remand, the district court looked to the '215 parent patent specification and construed "instructions." The Federal Circuit affirmed the district court's construction, in large part by looking to the specification. *Nazomi 2002 IV*, 266 F. App'x at 940. The issue before this court is the same because the patents share common disclosures. The '362 patent's specification and drawings are essentially identical to those in the '215 parent patent. The '436 patent contains all of the drawings of the '215 parent patent and much of the specification, plus additional material.

Nazomi objects that there are significant differences between the claims of the patents that suggest different meanings and thus the issues are not identical. Although it is true that the claims of these patents differ, the language in the claims must be understood in light of the teachings of the specifications, which are essentially identical. The specification should be the source of the definition of unclear terms in the claims, not a general dictionary definition such as Nazomi proposes. *See Phillips*, 415 F.3d at 1320-21. Because all of the patents share a common specification and the "specification is the single best guide to the meaning of a disputed term," *id.* at 1321 (internal quotations omitted), the claim construction issue before this court is identical to the one already decided in *Nazomi 2002*. Therefore, collateral estoppel applies.

3. Stack-Based Instructions Must Be Translated

The court must interpret a term "with the full understanding of what the inventors actually invented and intended to envelop with the claim." *Phillips*, 415 F.3d at 1316. Even where the claims of a patent might otherwise imply a broader claimed invention, if the specification as a whole defines the invention more narrowly, it limits the claims to that narrower definition. *See C.R. Bard, Inc. v. U.S. Surgical Corp.*, 388 F.3d 858, 863 (Fed. Cir. 2004). The summary of the invention is a good place to look for such limiting statements on the patent as a whole. *Id.* Generally, the description of a single embodiment in the specification should not alone limit broad claim language. *See Absolute Software, Inc. v. Stealth Signal, Inc.*, 659 F. 3d 1121 (Fed. Cir. 2011); *Osteonics Corp. v. Wright Med. Tech., Inc.*, 540 F.3d 1337, 1345 (Fed. Cir. 2008). However, where the specification describes an embodiment as the "primary objective" of the invention, describes it as the "present invention," states that it *is* the invention, or otherwise clearly and consistently uses a term in a more limited manner, the specification can limit otherwise broad language. *See id.*; *Verizon Services v. Vonage Holdings*, 503 F.3d 1295, 1308 (Fed. Cir. 2007); *Alloc, Inc. v. Int'l Trade Comm'n*, 342 F.3d 1361, 1368 (Fed. Cir. 2003).

This case is similar to *Verizon Services v. Vonage Holdings*, where the Federal Circuit limited the scope of the patent to how it described the invention in the Disclosure of the Invention section. 503 F.3d at 1308 (Fed. Cir. 2007). The court held that where a patent "describes the features of the 'present invention' as a whole, [the] description limits the scope of the invention." *Id.* Similarly, in *C.R. Bard v. U.S. Surgical*, the court found that the summary of the invention and the rest of the specification unequivocally defined the patent more narrowly than the claim language otherwise indicated, thus requiring the narrower claim construction. *C.R. Bard*, 388 F.3d at 863-66.

Similar to *Verizon* and *C.R. Bard*, the specifications of the '362 and '436 patent clearly limit the scope of the claimed invention to executing stack-based instructions by translating them into register-based instructions. First, the summaries of both patents contain a description of the "present invention," which requires translating bytecodes into native instructions. '362 patent col.2 ll.7-8; '436 patent col.2 ll.6-8. The summaries emphasize that the primary advantage of the patented invention over the prior art is to "translate Java™ bytecodes into native instructions." '362 patent at col.2 ll.14; '436 patent at col.2 ll.7-8.

Second, the rest of the patents' specifications so consistently describe the invention as a hardware apparatus used to translate stack-based instructions into register-based instructions that the inventor could not be said to have intended to claim anything else. The '362 patent repeatedly refers to its invention and purpose as translating stack-based instructions into register-based native instructions. *See, e.g.*, '362 col.2 ll.6-8 ("The present invention generally relates to a Java™ hardware accelerator which can be used to quickly translate bytecodes into native instructions."); *id.* at col.2 ll.25-28 ("[T]he hardware accelerators of the present invention are not limited for use with Java™ language and can be used with any stack-based language that is to be converted to register-based native instructions."); *id.* at col.4 ll.8-9 ("The instructions translations unit is used to convert Java bytecodes to native instructions."); *id.* at col.5 ll.64-65 ("The Java translating machine translates the Java bytecode into a native instruction."). Similarly, the '436 patent, which covers the same general invention, also clearly describes the patent as translating stack- to register-based instructions. *See, e.g.,* '436 Patent at col.2 ll.6-8 ("The present invention generally relates to Java hardware accelerators used to translate Java bytecodes into native instructions."); *id.* col.2 ll.19-22 ("Another embodiment of the present invention comprises a hardware accelerator to convert stack-based instructions into register-based instructions."); *id.* col.2 ll.43-47 ("Yet another embodiment of the present invention comprises a hardware accelerator operably connected to a central processing unit, the hardware accelerator adapted to convert stack-based instructions into register-based instructions.").

4. Nazomi's "Some or All" Argument

Nazomi argues that the patent is not limited to translating stack-based instructions to register-based instructions because language in the '362 and '436 patents describe the invention as capable of doing "some or all" of a list of tasks which includes "translating bytecodes to native instructions." '362 patent col.3 ll.10-19; '436 patent col.4 ll.1-15. In other words, according to Nazomi, the patented hardware accelerator is not necessarily required to be able to translate bytecodes to native instructions. However, such an interpretation of the "some or all" language is nonsensical and ignores what the patent is all about. If the hardware accelerator does not do translating of bytecodes to native instructions, it would not relate to what the "SUMMARY OF THE INVENTION" says the invention relates to. "The present invention generally relates to a Java hardware accelerator which can be used to quickly translate Java bytecodes into native instructions." *Id.* at col.2 ll.6-8.

The reasonable meaning of the list of tasks of which the invention can do "some or all" is that it can do the task of translating bytecodes to native instructions and may also be able to do some or all of the other listed tasks.

5. Construction of "Instructions"

For the reasons stated above, the court construes "instructions" as: "either stack-based instructions that are to be translated into register-based instructions, or register-based instructions that are input to the CPU pipeline." In either case the "instructions" must be upstream of the decode stage of the CPU pipeline. As used in the claims of the patent, "instructions" cannot mean the control signals that are the output of the decode stage.

B. "Processing the Stack-Based Instructions including Generating a Second Output" / "Second Output"

Nazomi's Proposed Construction	Defendants' Proposed Construction
Processing the stack-based instructions including generating a second output: Receiving stack-based instructions and generating a decoded output that is input to the execution unit. **Second output:** a decoded output that is input to the execute unit when stack-based instructions are processed in a hardware accelerator mode.	**Processing the stack-based instructions including generating a second output:** The stack-based instructions are translated into register-based instructions (the second output). **Second output:** register-based instructions.

The parties ask the court to construe the limitation in claim 1 of the '362 patent that states "processing the stack-based instruction including generating a second output" and, in particular, "second output." '362 patent col.7 ll.54-55. As Nazomi stated in its reply brief, the "primary dispute between the parties with regards to this claim element is essentially the same dispute with regard to 'instruction.'" Pl.'s Reply at 10. Because the processing of stack-based instructions requires translating them into register-based instructions, *see* Part III.A.3 *supra*, and for the reasons explained below, the "processing" described in the phrase "processing the stack-based instruction including generating a second output," must include: (1) the translation of the stack-based instructions into register-based instructions, and (2) the fetch and decode of the register-based instructions to convert them into control signals.

Claim 1 of the '362 patent recites two parallel limitations, the second of which is at issue in this construction. '362 patent col.7 ll.49-58. Because of the similarity of language in the two limitations, the first limitation is essential to understanding the second. The claim requires:

> 1. "processing the *register-based* instructions including generating a *first output*, and processing the first output in an execution unit using the data from the first register file"
>
> 2. "processing *the stack-based* instructions including generating a *second output*, and processing the second output in the execution unit using the operands from the first register file."

Id. (emphasis added). The key to construing the claim is the first and second outputs, which are the inputs to the execution unit. The court finds that the first and second outputs must be of the same type, otherwise the execution unit would be required to take multiple types of output as an input, which would require needlessly redundant hardware.

The input to the execution unit is either processed register-based instructions (first output) or processed stack-based instructions (second output). The only processing of register-based instructions discussed in the patent is the decoding that converts register-based instructions into control signals. Therefore, the first output must be control signals.

On the other hand, the patent teaches that stack-based instructions undergo two types of processing. First, they are translated into register-based instructions, and second those register-based instructions are decoded into control signals. Because, as explained below, the outputs (also the inputs to the execution unit) must be the same, the second output must also be control signals.

If the second output were register-based instructions, then the execution unit would have to include a decode unit to process those instructions into control signals. However, according to the first claim limitation, the decoding of register-based instructions in the decode unit happens outside of the execution unit. If the second output were register-based instructions, there would be two identical decode units right next to each other, one immediately outside the execution unit and another immediately inside. Thus, the only reasonable construction is that the second output is control signals.

Therefore, the "second output" is construed as "control signals." And "processing the stack-based instruction including generating a second output" is construed to mean: "the processing of stack-based instructions by translating them into register-based instructions and then decoding them into control signals."

C. "Execution Unit"

Nazomi's Proposed Construction	Defendants' Proposed Construction
Hardware capable of executing instructions, including arithmetic or logical instructions, by processing a decoded input.	Claim 1: The execute logic of the execute state of the CPU Claim 48: The execute logic of the execute state of the CPU and the circuitry which translates stack-based instructions into register-based instructions and the register-based instruction fetch and decode units.

The patent only uses the term "execution unit" in the claims and not elsewhere in the written description. The parties agree that claim 1 uses "execution unit" synonymously with the "execute logic." *See* Pl.'s Br. 15; Defs.' Br. 21. The "execute logic" is described in the '362 patent and is the hardware that takes decoded instructions (control signals) as input and performs the "execute" phase of the CPU pipeline. '362 patent col.5 ll.20-25. In apparent contrast, claim 48 describes an "execution unit" that executes stack- and register-based *instructions* rather than control signals. '362 patent col.10 ll.59-62. Because the control signals themselves are not actually instructions, the "execution unit" in claim 48 is either not the execute logic as the parties agree it is in claim 1, or claim 48 uses "executes" indirectly such that when it claims an "execution unit to execute instructions" it means that the "execution unit" executes the downstream result of processed instructions, which are control signals.

1. Other Claims

Claims 27, 74, 72, and 87 also use the term "execution unit," although none of the claims is at issue in this litigation. They do provide, however, some guidance as to the meaning of the term. The language of independent claims 27 and 74 support construing "execution unit" as the execute logic. Both claims describe limitations wherein the execution unit processes *decoded* instructions, which are control signals. *See* '362 patent col.9 ll.2-20, col.12 ll.29-47. Claims 1, 27, and 74 all indicate that instructions must be processed before they can be "executed" in the "execution unit," which is consistent with the execution unit being the execute logic, which takes control signals as input.

Claim 72, dependent on claim 48, provides some support for Nazomi's indirect execution argument. It adds a limitation describing possible upstream processing of an instruction before the instruction is executed in the execution unit. *Id.* at col.12 ll.21-26.

Finally, independent claim 87 uses language that mirrors the relevant language from claim 48, but does not otherwise provide any guidance on how to construe the term. *See* '362 patent col.13 ll.45-48 (claiming an "execution unit to process instructions of a plurality of instructions [sic] sets including a register-based instruction set and a stack-based instruction set.").

2. Reconciling the Different Uses

Courts should construe terms consistently throughout claims whenever possible. *See Callicrate v. Wadsworth Mfg., Inc.*, 427 F.3d 1361, 1371 (Fed. Cir. 2005). The '362 patent specification, a Federal Circuit decision in *Nazomi 2002*, and the prosecution history provide support for Nazomi's indirect execution argument, which is the only way to construe the term "execution unit" consistently throughout the claim.

First, the specification describes the execute logic as executing "native instructions." '362 patent col.5 l.23. This description comes immediately after explaining how the execute logic is downstream of the instruction decode, which converts instructions into control signals. At least in this context, the patent uses "execute" indirectly. Similarly, in the section describing prior art conversion of Java bytecodes using software, the patent explains that to "execute a Java™ program, a bytecode interpreter takes the Java™ bytecodes, converts them to equivalent native processor instructions and *executes* the Java™ program." '362 patent col.1 ll.28-32. The patent must be referring to indirect execution because the bytecode interpreter is software and thus does not literally execute a program. The Federal Circuit made this same point about "execution" by the bytecode interpreter in relation to the '215 parent patent. *Nazomi 2002 IV*, 266 Fed. Appx. at 940-41.

Second, the Federal Circuit also held that the patentee used execute indirectly in a different claim. The court construed "execute" as used in claim 30 of the '215 parent patent to "refer only to indirect causation." *Nazomi 2002 IV*, 266 Fed. Appx. at 940. Claim 30 claimed a central processing unit that included "an execution unit to execute the register-based instructions."

'215 parent patent col.8 ll.60-62. On remand, the district court said about the same issue that the "inventor must have been referring to this downstream effect rather than indicating that native instructions exist after the decode stage." *Nazomi 2002 III*, 2006 WL 2578374, *6 (N.D. Cal. Sept. 6, 2006). The Federal Circuit and district court's construction in *Nazomi 2002* is consistent with Nazomi's proposed construction of "execution unit" as the execute logic, which executes instructions indirectly.

Finally, the prosecution history supports Nazomi's argument that the patent uses "executes" indirectly in this context. In the prosecution history, the examiner questioned whether the patent described the limitation that stack- and register-based instructions were executed in one "execution unit." Anderson Decl., Ex. 6 at 30, Dkt. No. 409-8. The applicant's response points the examiner to Figure 3 and argues that the execute logic (which can receive only control signals) executes stack- and register-based instructions because they are processed upstream of the execute logic. *Id.* For stack-based instructions, the applicant specifically points to accelerator 42 (an independent hardware unit upstream of the execute logic), which translates stack-based instructions into register-based instructions, to explain how the execute logic can execute stack-based instructions. *Id.* The applicant concludes that, therefore, it is possible for stack- and register-based instructions to be executed in the "execution unit." *Id.* This passage indicates that the inventor was using the word "execute" indirectly. It also supports the argument that the execution of stack-based instructions necessarily first requires translation into register-based instructions as it is the only evidence provided by the applicant of how its claimed invention could execute stack- and register-based instructions.

3. Construction of "Execution Unit"

The court construes "execution unit" as: "the execute logic." However, based upon the court's previous constructions and the arguments supporting the "execute logic" construction, the court also holds that the execution of *instructions* in the execution unit refers to indirect execution. Indirect execution requires circuitry which: (1) translates stack-based instructions into register-based instructions and (2) fetches and decodes register-based instructions into control signals that are the input to the execution unit. As a practical matter in light of the construction of "instructions," there is no substantive difference between the parties' different constructions of "execution unit."

D. "Hardware Accelerator to Process Stack-Based Instructions"

Nazomi's Proposed Construction	Defendants' Proposed Construction
Hardware capable of accelerating the processing of stack-based instructions.	Circuitry which can be used to translate stack-based instructions into native instructions.

The primary dispute between the parties' constructions is whether the hardware accelerator requires circuitry to translate stack-based instructions into native instructions. *See* Pl.'s Br. at 20. As already decided above, the patents claim accelerating the execution of stack-based instructions by translating them into register-based instructions in hardware. Therefore, the term refers to "circuitry which can be used to translate stack-based instructions into register-based instructions."

Throughout the patent, the hardware accelerator is described as translating stack-based instructions to register-based instructions. The summary of the invention defines it as relating to "a Java™ hardware accelerator which can be used to quickly translate bytecodes into native instructions." '362 patent col.2 ll.6-8. Similarly, the '436 patent's abstract summarizes the hardware accelerator as "enabled to convert the Java™ bytecodes into native instructions." '436 patent. Similar language appears throughout the patents, establishing that the invention, hardware acceleration of stack-based instructions, is limited to accelerating by translating stack-based instructions into native, register-based instructions. *See* Section A.3 *supra*.

Nazomi argues that claim differentiation requires its proposed construction be adopted. The Federal Circuit, however, has held that claim differentiation is "not a hard and fast rule and will be overcome by a contrary construction dictated by the written description or prosecution history." *Seachange Int'l, Inc. v. C–COR, Inc.*, 413 F.3d 1361, 1369 (Fed. Cir. 2005). Where, as here, there is arguably a conflict between the teachings of the specification and the doctrine of claim differentiation, the teachings of the specification control. *See id.* Claim differentiation cannot render a claim broader than that which the inventor disclosed as the invention in the specification, and claims "written in different words may ultimately cover substantially the same subject matter." *Id.* (citing *Multiform Desiccants, Inc. v. Medzam, Ltd.*, 133 F.3d 1473, 1480 (Fed. Cir. 1998). Dependent claim 4 of the '436 patent adds the limitation to claim 2 that "the hardware accelerator processes the stack-based instructions in cooperation with the execute logic by converting the stack-based instructions into register-based instructions for execution in the execute logic." If this claim is read to suggest that translation did not necessarily happen in claim 2, which recites "a hardware accelerator to process stack-based instructions," this interpretation would contradict the other claims and all of the disclosures in the '362 and '436 patent, which demonstrate that the processing of stack-based instructions requires translation. The teachings of the specification must control over the doctrine of claim differentiation.

Therefore, "hardware accelerator to process stack-based instructions" is construed as "circuitry which can be used to translate stack-based instructions into native instructions."

E. "Stack-Based Instructions"

Nazomi's Proposed Construction	Defendants' Proposed Construction
Instructions that manipulate operands from an operand stack.	An instruction that instructs the processor to manipulate a push-down operand stack and that does not refer to a register index of a register file.

There is little difference between the parties' proposed constructions. Defendants insist that their definition reflects two fundamental differences between stacks and registers, which Nazomi's definition does not reflect. The court generally agrees with defendants' understanding of "stack-based instructions," but finds that Nazomi's construction reflects it.

First, defendants' definition specifies that the operand stack is a "push-down" stack. To the extent defendants use "push-down" to reflect the last-in, first-out nature of stacks, the court agrees, but the term "push-down" is not used anywhere in the patent. As part of his description of Java, Judge Fogel explained stack-based memory as storing information "on a last-in, first-out basis." *Nazomi 2002 III*, 2006 WL 2578374 at *1. Defendants used the phrase "last-in, first out" as part of their tutorial and while Nazomi did not use that exact language, it implied the same thing. Although "operand stack" is probably sufficiently descriptive, for clarity, the court will add the modifier "last-in, first-out" to its construction.

Second, defendants' claim that their construction is superior because it reflects the difference between stack- and register-based instructions by noting that stack-based instructions do not refer to registers. Nazomi, however, objects to defendants' construction to the extent defendants would use their construction to preclude implementation of an operand stack in a register file, which is something the '362 patent teaches. Of course, implementing the operand stack in a register file requires translating the stack-based instructions into register-based instructions so that they can actually refer to the registers. Defendants are correct that by definition, stack-based instructions do not refer to registers, but the court finds that Nazomi's simpler construction is adequate to reflect this fact.

Therefore, the court construes "stack-based instructions" as "instructions that manipulate operands from a last-in, first-out operand stack."

IV. ORDER

For the reasons set forth above, the court construes the claims as follows:

Claim Terms	Construction
instructions	either stack-based instructions that to be translated into a register-based instructions, or register-based instructions that are input to the CPU pipeline. In either case the "instructions" must be upstream of the decode stage of the CPU pipeline. As used in the claims of the patent, "instruction" cannot mean the control signals that are the output of the decode stage
second output processing the stack-based instruction including generating a second output	control signals processing of stack-based instructions by translating them into register-based instructions and then decoding them into control signals
execution unit	the execute logic. However, the execution of *instructions* in the execution unit refers to indirect execution. Indirect execution requires circuitry which: (1) translates stack-based instructions into register-based instructions and (2) fetches and decodes register-based instructions into control signals that are the input to the execution

	unit.
hardware accelerator to process stack-based instructions	circuitry, which can be used to translate stack-based instructions into native instructions
stack-based instructions	instructions that manipulate operands from a last-in, first-out operand stack.

Dated: June 14, 2013

Ronald M. Whyte

Ronald M. Whyte
United States District Court Judge